MANY-VISITS VEHICLE ROUTING PROBLEMS

Jayme L. Szwarcfiter

NCE-13/88

Outubro/88

Universidade Federal do Rio de Janeiro
Núcleo de Computação Eletrônica
Caixa Postal 2324
20001 - Rio de Janeiro - RJ
BRASIL

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
NÚCLEO DE COMPUTAÇÃO ELETRÔNICA

ABSTRACT

We consider constrained routing problems where each city is to be visited possibly many times. Two algorithms are given having complexities which are exponentials in the number of cities, but not in the number of visits. In addition, a criteria is proposed for classifying algorithms for general many-visits routing/scheduling problems

RESUMO

Consideramos problemas de roteamento com restrições, onde cada cidade deve ser visitada, possivelmente, várias vezes. Descrevemos dois algoritmos cujas complexidades são expressões exponenciais no número de cidades, porém não no número de visitas. Além disso, propomos um critério de classificação de algoritmos para problemas gerais de scheduling/roteamento com múltiplas visitas.

1. INTRODUCTION

We consider routing problems having many objects of the same type. The objects correspond to cities and vehicles (or alternatively jobs and machines when scheduling problems are considered). Objects of the same type are those having identical descriptions. For example, a set of jobs having the same processing times, resource requirements, release dates and so on. Altogether, there are n objects distributed into t types. The objective is to find algorithms for solving the routing (scheduling) problems in general, such that their complexities are polynomials in n (if possible, log n) and perhaps exponentials in t.

The interest of looking at these problems is clearly when $n >> t$. There is a practical motivation in this case, namely routing problems which require each city to be visited possibly many times. In addition, a theoretical motivation: can a given many-visits problem be solved by an algorithm which is polynomial (or logarithmic) in the number of visits?

In the next section we mention some existing algorithms for different problems of this kind. In Section 3 we describe a general many-visits vehicle routing problem (VRP). Sections 4 and 5 are ILP and dynamic programming formulations of the VRP, respectively. Its actual solution is obtained in Section 6. Section 7 describes a criteria for classyfing many-visits problem, based on the complexities of their algorithms. The conclusions of the last section include comments on a known problem of this kind, namely the aircraft landing scheduling problem.

2. SOME EXISTING ALGORITHMS

The first problem to be considered is that of minimizing the length of a multiprocessor schedule. That is, there are n jobs of t different types (i.e. no more than t jobs have different processing times) and m identical processors. L denotes the largest processing time. The aim is to find a schedule having minimum

makespan. Leung [4] described an algorithm requiring time $O(n^{2(t-1)}(\log m)(\log L))$ and space $O(n^{t-1}\log m)$. Blazewicz, Kubiak and Szwarcfiter [1] solve this same problem using $O(n^{2t}\log m)$ time and $O(n^{t}\log m)$ space. Both algorithms employ dynamic programming and can be generalized with no difficulty to unrelated processors. The algorithm [4] is however restricted to jobs with integer lengths, whereas in [1] the lengths might be real numbers. If the number of processors is also fixed, [1] presented a different algorithm based on ILP having time complexity $O(n + (\log(n+L))^{c})$, where c is a constant.

Consider now the problem of minimizing the length of a schedule of a job shop problem with unit time operations, resource constraints and release dates. The following variables are fixed: the maximum number of operations per job, the total number of resources, number of processors and maximum resource requirements per operation. The arbitrary quantities are the number of jobs, quantity of each resource and the release date for each job. In fact, this formulation does not correspond strictly to a many-visits problem, because the release dates are arbitrary. However, it can be solved by applying techniques similar to those used in the solution of some many-visits problems as shown in [6] where a polynomial time dynamic programming algorithm is described.

Psaraftis [5] presented an algorithm for solving some sequencing/routing many-visits problems with a fairly general cost function. This algorithm has complexity $O(t^{2}(n+1)^{t})$, for a total of n jobs of t types and is based on dynamic programming.

Cosmadakis and Papadimitriou [2] described an algorithm for solving the many-visits traveling salesman problem (TSP). Its method differs from the other known many-visits algorithms in the sense that besides dynamic programming it employs other techniques for reducing the time complexity of the algorithm. Using graph

theoretical concepts the algorithm constructs eulerian digraphs, searching for its corresponding indegree sequences. This step is performed efficiently through a good characterization of these sequences. Basically, if there are n cities of t types then instead of considering all possible sequences of types the algorithm only generates those which are indegree sequences of minimal eulerian digraphs. The recognition of such sequences is done using a graph theoretical characterization. The overall complexity of the algorithm is exponential in t, but logarithmic in n.

3. MORE GENERAL ROUTING PROBLEMS

Consider a set of cities $C=\{C_1,\ldots,C_t\}$, in which each C_i is to be visited C_i times. $F=\{F_1,\ldots,F_m\}$, $m<t$, denotes a set of fleets, each F_k formed by f_k identical vehicles having a weight capacity W_k and a distance capacity D_k, $1\leqslant k\leqslant m$. In each visit to C_i we have to deliver a weight w_{ik}, where F_k is the fleet of the vehicle used in the visit. For each $1\leqslant i,j\leqslant t$ and $1\leqslant k\leqslant m$ the symbol d_{ijk} denotes the distance from C_i to C_j, when using a vehicle of F_k. For $1\leqslant k\leqslant m$, C_k is called the depot of F_k and satisfies $c_k=f_k$. Also, $d_{ijk}=\infty$ for any k, whenever $1\leqslant i,j\leqslant m$.

An open subtour of F_k is a sequence of cities with repetitions allowed, visited by a vehicle of F_k. A subtour of F_k is such a sequence in which the first and last city coincide. A tour of F_k is a set of f_k subtours of F_k such that:

(i) each subtour contains C_k exactly once and no other depot,

(ii) the sum of the weights w_{ik} over all cities C_i in the subtours is $\leq W_k$, and

(iii) the sum of the distances d_{ijk} between all pairs C_i, C_j of consecutive cities in the subtours is $\leq D_k$.

An optimal tour is a set of tours $T=\{T_1,\ldots,T_m\}$ such that:

(i) each C_i appears exactly c_i times in T, and

(ii) the sum of all distances in T is minimum.

The many-visits VRP consists in finding an optimal tour, given the cities, fleets and all costs involved. We consider the number of cities as a fixed variable, whereas the number of visits to each city, number of vehicles of each fleet as well as the values of all weigths, distances, weigth capacities and distance capacities are all arbitrary. We describe two algorithms for solving this VRP. The motivation in studying this case is when the number of visits (vehicles) is much larger than that of cities (fleets).

4. AN ILP FORMULATION

Let x_{ijk} denote the number of times C_i proceeds C_j in a subtour of F_k. The above VRP can be solved by the following ILP.

$$\min \sum_{\substack{1\leq i,j\leq t \\ 1\leq k\leq m}} x_{ijk}\, d_{ijk} \qquad \ldots. (1)$$

$$\text{s.t.} \quad \sum_{1\leq j\leq t} x_{ijk} = \sum_{1\leq j\leq t} x_{jik},$$

$$\text{for } 1\leq i\leq t \text{ and } 1\leq k\leq m \qquad \ldots. (2)$$

$$\sum_{\substack{1\leqslant j\leqslant t\\ 1\leqslant k\leqslant m}} x_{ijk} = c_i, \qquad \text{for } 1\leqslant i\leqslant t \qquad \ldots. (3)$$

$$\sum_{1\leqslant j\leqslant t} x_{kjk} = c_k, \qquad \text{for } 1\leqslant k\leqslant m \qquad \ldots. (4)$$

$$\sum_{1\leqslant i,j\leqslant t} x_{ijk}\, w_{ik} \leqslant W_k \qquad \text{for } 1\leqslant k\leqslant m \qquad \ldots. (5)$$

$$\sum_{1\leqslant i,j\leqslant t} x_{ijk}\, d_{ijk} \leqslant D_k, \qquad \text{for } 1\leqslant k\leqslant m \qquad \ldots. (6)$$

$$\sum_{i,j\in S} x_{ijk} < \sum_{\substack{i\in S\\ 1\leqslant j\leqslant t}} x_{ijk}, \qquad \text{for } S \subset \{1,\ldots,t\} - \{k\} \text{ and } 1\leqslant k\leqslant m \qquad \ldots. (7)$$

Constraints (2) asure that whenever we enter a city using a vehicle of F_k we can always leave it with the same vehicle. Because of (3) it is guaranteed that each C_i is visited exactly c_i times, whereas (4) asure that a depot C_k is visited only by vehicles of F_k. The weight and distance constraints of the fleets are guaranteed by (5) and (6), respectively, and (7) describe the subtour constraints. As usual with ILP formulations of routing problems, a central point is the handling of subtours. The constraints (7) do not eliminate necessarily the unwanted ones. Instead they are afterwards properly transformed, while maintaining the total distance used.

According to Lenstra's algorithm [3] the above ILP can be solved in time polylog in the total number of visits and exponential in the number of cities.

5. A DYNAMIC PROGRAMMING FORMULATION

In order to describe a dynamic programming solution for the VRP stated in Section 3 we need the following additional notation.

Let $C=\{C_1,\ldots,C_t\}$ be the set of cities, c_i being the number of visits required for C_i. The profile of C is the vector $C_\#=(c_1,\ldots,c_t)$, while a subprofile of $C_\#$ is $C'_\#=(c'_1,\ldots,c'_t)$, where $c'_i \leqslant c_i$, $1 \leqslant i \leqslant t$. Define $C_o=(o,\ldots,o)$, $C'_\# \pm i = (c'_1,\ldots,c'_{i-1},c'_i \pm 1,c'_{i+1},\ldots,c_t)$ and $C'_\# - C''_\# = (c'_1-c''_1,\ldots,c'_t-c''_t)$.

Let $y(C'_\#,F')$ denote the minimum length of an optimal tour of a VRP having profile $C'_\#$ and fleet set F'. In order to obtain the solution $y(C_\#,F)$ of the VRP we use the following equation.

$$y(C'_\#,F') = \min_{\substack{C''_\# \subset C'_\# \\ F_k \in F'}} \{y(C''_\#, F'-\{F_k\}) + y(C'_\#-C''_\#, \{F_k\})\}$$

The initial conditions correspond to the cases where $C'_\#=C_o$ or $|F'|=1$. They can be computed as follows

$$y(C_o,F') = 0$$

$$y(C'_\#, \{F_k\}) = \begin{cases} \infty, & \text{if any among (8)-(12) below holds} \\ z(C'_\#, \{F_k\}, k), & \text{otherwise,} \end{cases}$$

where $c'_k \neq f_k$ .... (8)

$c'_j \neq 0$ for $1 \leqslant j \leqslant m$ and $j \neq k$ .... (9)

$$\sum_{m<j\leqslant t} c'_j < f_k \quad \text{... (10)}$$

$$\sum_{1\leqslant j\leqslant t} w_{jk} c'_j < W_k \quad \text{... (11)}$$

$$\sum_{1\leqslant j,\ell\leqslant t} d_{j\ell k} c'_j < D_k \quad \text{... (12)}$$

and $z(C'_{\#}, \{F_k\}, i)$ denotes the minimum length of an open subtour of F_k having subprofile $C'_{\#}$, starting at city C_k and ending at C_i. Its value can be obtained by computing

$$z(C'_{\#}, \{F_k\}, i) = \min_j \{z(C_{\#} - i, \{F_k\}, j) + d_{jik}\},$$

with the initial condition

$$z(C_0 + k + i, \{F_k\}, i) = d_{kik}$$

The constraint (8) above described asures that the number of depots in each tour of F_k is correct, while (9) guarantees that the vehicles of F_k do not visit depots of other fleets. Inequality (9) asures that all vehicles of F_k are used, while (9) and (10) are the weight and distance constraints, respectively.

In order to compute the final value $y(C_{\#}, F)$, we need to compute all $z(C'_{\#}, \{F_k\}, i)$ and all intermediate $y(C'_{\#}, F')$. By a simple counting we can conclude that the algorithm requires exponential

time in the number t of cities, but polynomial time in the number of visits.

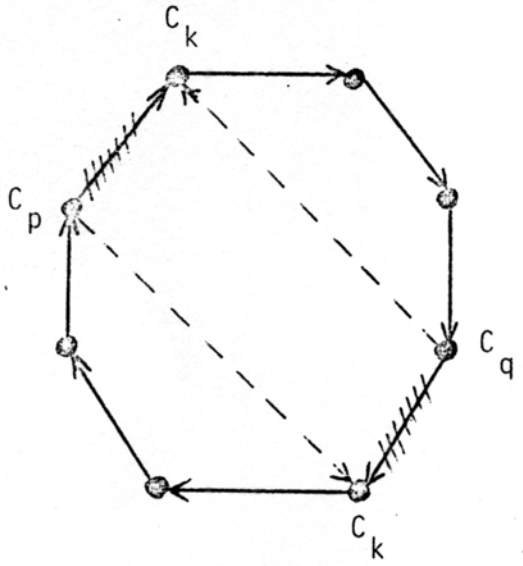
6. FINDING THE OPTIMAL TOUR

Both the ILP and dynamic programming methods described for solving the VRP of Section 3 do not construct the actual solution. Instead they inform how many times city C_i proceeds C_j in an optimal tour. Using this information it is straightforward to construct a set of m tours, one for each fleet in which the total number of visits is correct. To adjust the number of subtours in each tour to make it equal the number of vehicles of the corresponding fleet, we can use the operations below described.

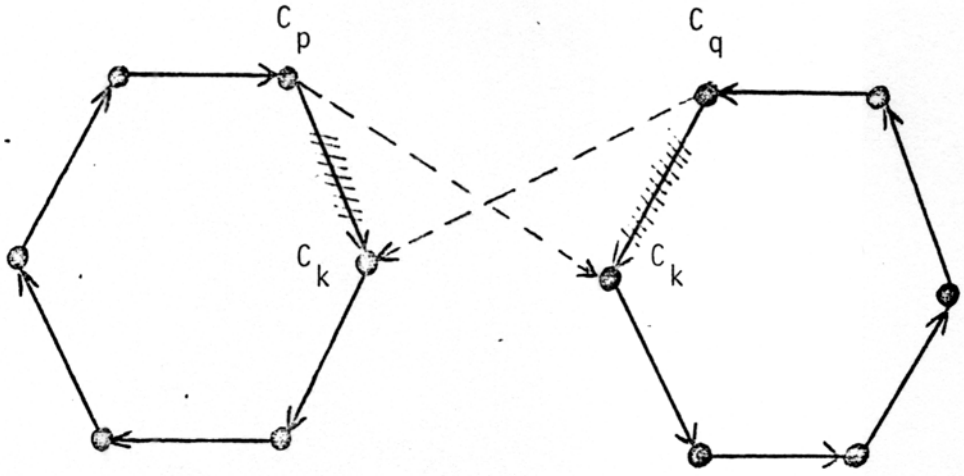
The split operation considers a subtour in which there exists some city C_k twice, although not in consecutive positions. Let C_p and C_q be the cities preceeding C_k in the first and second visit of C_k in the considered subtour, according to some arbitrary order. If we now choose to divert from C_p directly to the second instance of C_k instead, and from C_p to the first visit of C_k then we split the subtour into two others whose sum of their lengths equals that of the previous subtours.

The join operation is the inverse of the above. Now we have two distinct subtours, both visiting city C_k. Using similar arguments as for the split operation we can conclude that we can always replace these two subtours by a single one performing exactly the same visits and preserving the total length of the subtours.

For any fleet F_k if the number of obtained subtours is less than its corresponding number f_k of vehicles then the constraints



(a) Split Operation



(b) Join Operation

Figure 1

imposed by the algorithms asure that there exists necessarily a subtour visiting the depot C_k twice. Therefore we can perform the split operation to increase the number of vehicles used. Similarly, if there are more subtours than vehicles then the constraints now guarantee that there will be two distinct subtours visiting a same city and hence the join operation can be applied.

7. CLASSIFICATION AND OPEN PROBLEMS

Clearly, the input for a many-visits problem may consists of individual descriptions of the n objects of the problem or otherwise we can describe each distinct object just once and specify the number of copies of it. In the first case, the length of the input is $\Omega(n)$ and in the second $\Omega(t+\log k)$, where k is the maximum number of copies among the objects and t the number of types. This means that an algorithm which is polynomial according to the first input may turn exponential, using the second. Of course, it would be fair to consider as a measure of complexity this second choice of input. On the other hand, the final output in general requires information to be given for each of the n objects and therefore is $\Omega(n)$. The following definitions would then be useful to handle these problems.

Let P be a many-visits problem having n objects of t types, input $O(t+\log n)$ and output $O(n)$. A weakly polynomial time algorith for P is one whose complexity is polynomial in n and exponential in t. An algorithm is of trully polynomial time when it can be divided into two distinct and consecutive phases. The first one is an algorithm whose complexity is logarithmic in n and exponential in k, while the second is polynomial in n, but independent of k. Since in many-visits problems the output may be long compared to the input, when adopting the short input as standard, the idea is to separate the main computation from a simple editing of the output, for instance.

According to the above classification, the algorithms by Cosmadakis and Papadimitriou [2], the ILP for the VRP described in

Section 4 and the ILP of [1] for minimizing the length of a multiprocessor schedule are trully polynomial. However, it should be noted that the phase one of [2] is an algorithm which is linear with the input whereas that of the two other last mentioned ILP algorithms have time complexities which are polynomials of high degree with their input. The algorithms [4], [5] and the dynamic programming described in [1] and Section 5 are all of weakly polynomial time. The mentioned job shop scheduling algorithm [6] is actually of regular polynomial time, since there is no way of representing its input in length $O(t+\log n)$, because of the arbitrary release dates.

The most efficient approach to a many-visits problem is that introduced by Cosmadakis and Papadimitriou for the TSP, leading to a trully polynomial time with linear phase one algorithm. It can be shown that it is possible to extend this approach and maintain the nature of its complexity for the following two reductions of the VRP.

(1) The many-visits m-TSP with depot, that is a TSP having m salesmen all starting from a common depot and visiting each city possibly many times;

(2) The unconstrained many-visits VRP, that is a VRP as defined in Section 3, except that the constraints W_k and D_k are absent.

It would be interesting to know the answers of the following questions related to many-visits problems:

(1) Can the minimal eulerian subgraph technique of [2] be extended to the VRP, leading to a trully polynomial with linear phase one algorithm?

(2) Is there a trully polynomial time algorithm for minimizing the length a multiprocessor schedule with an arbitrary number of processors?

(3) Is there a weakly polynomial time algorithm for a VRP in which the weight and distance constraints apply to each individual vehicle, instead of each fleet?

(4) Is there a weakly polynomial time algorithm for the VRP with time windows (even if no weight or distance constraints are considered)?

8. CONCLUSIONS

We have considered scheduling/routing problems in which there are many copies of identical objects. A classification has been described which allows us to rank the complexity of algorithms for solving this kind of problems. In addition two algorithms have been formulated for the many-visits VRP. Such problems might be also of practical interest. Indeed a real application which falls into the category of many-visits problems has been described by Psaraftis [5], namely the scheduling of airplane landing. In this case, there are possibly many airplanes, but just a few types of them. The problem consists of finding a schedule for the landing of the airplanes in m runways which would minimize the sum of the landing times, or alternatively the time of the latest landing. Psaraftis solved this problem for m=1 or 2 and asked for the case m>2. The minimization of the total landing time is equivalent to the many-visits m-TSP and therefore can be solved efficiently by an extension of the algorithm by Cosmadakis and Papadimitriou, as mentioned in the previous section. On the other hand, minimizing the time of the latest landing corresponds to the bottleneck many-visits m-TSP. The latter is solvable by a variation of the ILP described in Section 4, for the many-visits VRP.

REFERENCES

1. J.Blazewicz, W.Kubiak and J.Szwarcfiter, Scheduling Independent Fixed-Type Taks, in R.Slowinski and J.Werglasz, eds., Advances in Project Scheduling, North-Holland, to appear;

2. S.S.Cosmadakis and C.H.Papadimitriou, The Traveling Salesman Problem with Many Visits to Few Cities, SIAM Journal on Computing 13(1984) 99-108;

3. H.W.Lenstra, Jr., Integer Programming with a Fixed Number of Variables, Mathematics for Operations Research 8(1983) 538-548;

4. J.Y-T.Leung, On Scheduling Independent Tasks with Restricted Executions Times, Operations Research 30(1982) 163-171;

5. H.Psaraftis, A Dynamic Programming Approach for Sequencing Groups of Identical Jobs, Operations Research 28(1980) 1347-1359;

6. J.L.Szwarcfiter, Job Shop Scheduling with Unit Time Operations under Resource Constraints and Release Dates, Discrete Applied Mathematics 18(1987) 227-233.